

Dynamic airspace sectorisation for flight-centric operations

Ingrid Gerdes, Annette Temme, Michael Schultz*
Institute of Flight Guidance, DLR Braunschweig, Germany

Abstract — Today's air traffic operations follow the paradigm of 'flow follows structure', which already limits the operational efficiency and punctuality of current air traffic movements. Therefore, we introduce the dynamic airspace sectorisation and consequently change this paradigm to the more appropriate approach of 'structure follows flow'. The dynamic airspace sectorisation allows an efficient allocation of scarce resources considering operational, economic and ecological constraints in both nominal and variable air traffic conditions. Our approach clusters traffic patterns and uses evolutionary algorithms for optimisation of the airspace, focusing on high capacity utilisation through flexible use of airspace, appropriate distribution of task load for air traffic controllers and fast adaptation to changed operational constraints. We thereby offer a solution for handling non-convex airspace boundaries and provide a proof of concept using current operational airspace structures and enabling a flight-centric air traffic management. We are confident that our developed dynamic airspace sectorisation significantly contributes to the challenges of future airspace by providing appropriate structures for future 4D aircraft trajectories, taking into account various operational aspects of air traffic such as temporally restricted areas, limited capacities, zones of convective weather or urban air mobility. Dynamic sectorisation is a key enabling technology in the achievement of the ambitious goals of Single European Sky and Flightpath 2050 through a reduction in coordination efforts, efficient resource allocation, reduced aircraft emissions, fewer detours, and minimisation of air traffic delays.

Keywords - airspace; dynamic sectorisation; controller task load; evolutionary algorithm; clustering

1 Introduction

The sectorisation of airspace considers requirements of air traffic management (safety, capacity and efficiency), users (unhindered access) and environment (restricted areas over cities, residential areas, etc.). Particularly, air traffic control requires the airspace to be structured in order to accommodate an appropriate operational infrastructure for airspace users and operators. The airspace sectorisation is primarily triggered by operational demands (such as handling of mixed traffic, balanced controller work load, procedure design or capacity management), territorial aspects (air sovereignty) and operational performance (Eurocontrol and FAA, 2016). In order to respond to the needs and future challenges of the air traffic (seamless European airspace), the European Commission (2004) initiated the Functional Airspace Block (FAB) design to implement multinational management and increase the airspace efficiency.

The dynamic airspace sectorisation presented in this paper deals with highly variable traffic patterns, which demand mid/short-term airspace adaptations. Our target is to create an appropriate, continuous airspace sectorisation in a fast and efficient way, which is able to react on current air traffic flows and efficiently support the airspace controller, even in uncommon traffic situations. Key drivers for dynamic traffic situations are characteristic air traffic flow patterns in Europe and the intercontinental connections (eastbound, westbound), as well as military actions (Islami et al., 2017), disruptive events such as volcanic ash eruptions (Luchkova et al., 2015), zones of convective weather (Kreuz, Luchkova and Schultz, 2016), prevention of contrails (Rosenow, Fricke and Schultz, 2017), consideration of commercial space operations (Kaltenhäuser et al., 2017) and integration of new entrants (Sunil et al., 2015; Temme and Helm, 2016), which demand an efficient operational solution.

Furthermore, our approach bridges the gap between structured and unstructured airspace designs and will also be a fundamental key element for an efficient management of the future urban airspaces. If, in the future, the urban area consists of a significant amount of movements of personal air vehicles, the frequency of traffic will follow the daily time-dependent demand for transportation. Similar to today, there is a clear indication of a highly used infrastructure, which changes over the day (e.g. morning and evening peaks in and out of the city). However, the urban airspace will also be limited and fragmented due to immanent safety aspects (e.g. minimum distance requirements) or restricted areas (e.g. critical infrastructure). The dynamic sectorisation provides a scalable approach to balancing the air traffic demands and operational requirements by clustering traffic patterns, identifying areas of high-density traffic and providing an efficient planning and control structure to support airspace operators and users. Air traffic management in urban areas is expected to be one of the most challenging tasks in the coming years. Instead of deriving future flow control methods from today's operations, we propose a fundamental change and provide an appropriate solution for handling dynamic traffic demands efficiently.

* Corresponding author.

E-mail addresses: ingrid.gerdes@dlr.de (I. Gerdes), annette.temme@dlr.de (A. Temme), michael.schultz@dlr.de (M. Schultz).

To ensure a more efficient allocation and a harmonised task load distribution, we consequently changed the current paradigm of traffic flow, which is determined by airspace structure ('flow follows structure'), to a dynamic approach of a structure that is adjusted to the traffic flow sequentially ('structure follows flow'). We contribute to the future flight-centric air traffic management with our specific approach of dynamically optimising the airspace focusing on the sector structure and resource allocation, considering both operational and economic efficiency targets (e.g. task load, fragmentation of flights by sectors). The approach of a dynamic reorganisation of airspace during the day of operation has several advantages, such as improved capacity utilisation through flexible use of airspace or appropriate distribution of task load for air traffic controllers and fast adaptation to changed operational boundaries. Dynamic Airspace Sectorisation (DAS) is a flexible method encompassing the idea of unstructured and (rigid) structured airspace, but differs from Dynamic Airspace Configuration (DAC), where predefined airspace blocks are combined to form new structures (merging and splitting). We are assuming a continuous airspace that will be separated without a specific demand for underlying structures but which considers both the current/future air traffic flows and the controller's ability to manage all assigned aircraft. DAS enables a continuous restructuring of airspace sectors that depends on current requirements, and will be an important element for efficient air traffic operations, taking into account both regular and disruptive events. In particular, DAS covers today's regular airspace structure and regional, trajectory-based, flight-/flow-centric operations challenged by SESAR Joint Undertaking (2015). As emphasised, we also see the DAS approach as a sustainable solution for future air traffic management in the context of urban flight operations. Our DAS solution can be adapted to different airspace management concepts and already covers all operational aspects needed for balanced handling of dynamic air traffic flow. It is scalable and therefore able to cope with a city's lower airspace (Choo and Yoon, 2018) as well as a huge part of the upper airspace (Schultz et al., 2017).

1.1 *Status quo*

Over the years, several authors have developed ideas for an automatic creation of airspace sectorisation or the adaptation of airspace sectors. Durand et al. (2015) provide a brief overview on metaheuristics used in airspace management. One of the first approaches using evolutionary algorithms was developed by Delahaye, Schoenauer and Alliot (1998). Other approaches applied the principles of graph theory, Voronoi diagrams, linear integer programming or clustering of flight tracks. A detailed overview of recent work is given by Sherali and Hill (2011) and Li et al. (2010). A comparison of eight different methods is provided by Zelinski and Lai (2011). Furthermore, a survey on the work carried out in the area of dynamic airspace sectorisation is given by Flener and Pearson (2013), listing relevant approaches of recent years in a systematic manner. Many of these ideas are of a more theoretical nature and not all are directly applicable to an authentic non-convex airspace problem.

1.2 *Objectives and document structure*

The objective of our research is to automatically provide an appropriate airspace sectorisation, comparable to today's sector structure, without incorporating explicit expertise of air traffic controllers. This is necessary for a more flight-centred approach to sectorisation where the structure of the sectorisation depends highly on the predicted traffic. Especially for new approaches, such as the idea of ecologically sensitive flight operations, where airspace shall be closed due to contrail management, it will be necessary to adapt the airspace to the changed requirements of the environment. Therefore, it is not possible to rely on controller knowledge due to variable and dynamically changing constraints. In our case, the controller behaviour is implicitly covered by a task load model. Furthermore, the aim of our approach is to efficiently consider varying air traffic flows in different time intervals and react on specific events which influence the air traffic system, such as zones of convective weather.

When developing the idea of automatic sectorisation (AutoSec), we had in mind both flight-centric operations and multi-criteria optimisation. AutoSec allows the creation of an appropriate airspace structure in a three-step approach: (1) fuzzy clustering of traffic flows on the day of operations, (2) generation of new sector structure based on Voronoi diagrams and (3) application of evolutionary algorithm in order to adjust and optimise the new sector structure depending on dynamic demands over the day of operations. Within this paper, we compare the actual sectorisation of a specific airspace with evolutionary optimised solutions, which are based in a first step on real airspace and in a second step on newly created airspace structure. The optimisation in both cases is carried out with the same evolutionary algorithm and evaluation function (Gerdes, Temme and Schultz, 2016).

The paper is structured as follows: In section 2, we provide the general background information referring to the different reorganisation types DAC and DAS, present the task load system used for the simulation and explain the structure of the simulation framework. Within section 3, the concepts of fuzzy clustering, Voronoi diagrams and evolutionary algorithms as well as the used data structure are

explained in general. The theoretical background and the necessary adaptations of the methods used to meet the problem of reorganising airspace are introduced in section 4. In particular, we provide a solution for handling non-convex sector shapes efficiently and demonstrate the concept of interim diagrams allowing a smooth transition between different sectorisations, which are caused by changing traffic flow patterns. The experimental setup is given in section 5. The results for the optimisation of the real sector structure of EDYYDUTA and our DAS approach, together with a comparison of these results, are presented in section 6. Furthermore, a first attempt at analysing the creation of building blocks is given. Finally, the conclusion provides an overview of the achieved results and an outlook on future research activities.

2 Background – optimisation approach and data structure

2.1 Approaches to airspace reorganisation

Approaches to reorganising airspace can generally be divided into two sub-groups, depending on type and target of partition: Dynamic Airspace Sectorisation and Dynamic Airspace Configuration. The latter includes all methods that rearrange predefined parts of the airspace in dependence on changing traffic patterns. For this, a basic set of airspace blocks is created and combined in changing combinations to meet the actual requirements. The target is to minimise the effort for the controllers when changing from one sectorisation to another and to increase the stability of the airspace sectorisation (Kopardekar, Bilimoria and Sridhar et al., 2007). Furthermore, this approach increases the familiarity of controllers with possible sectorisations because the airspace blocks are reused. Mehadhebi, K. (2000) e.g. used Voronoi diagrams to partition the French airspace, depending on the traffic density as constraint for the design of a route network. Tree search methods have been applied by Gianazza (2010) for airspace configuration from fixed airspace modules based on controller workload forecast. Bloem, M. and Gupta, P. (2010) adopted dynamic programming techniques in their DAC approach. In Jägare, Flener and Pearson (2013), constraint-based local search techniques with geometric and workload constraints built the basis for an approach generating a static airspace sectorisation from a regular mesh of cells.

In comparison, DAS approaches create a complete new sectorisation without considering actual airspace structure or using predefined elements, as is presented in Standfuß et al. (2018), Gerdes, Temme and Schultz (2016), Chen, Bi and Zhang (2013), Tang et al. (2012) and Basu et al. (2009). This reorganisation approach therefore creates a less familiar setting for controllers but can be used for situations where no basic knowledge concerning the best airspace structure is available. This is especially the case when the airspace sectorisation has to be adapted to uncommon or unfamiliar events, as is the case for special meteorological effects such as volcanic ash or airspace closed to prevent contrails. Examples for DAC are given by Sergeeva et al. (2015), Kulkarni, Ganesan and Sherry (2011) and Yangzhou and Defu (2014). Sergeeva et al. (2015) introduced a level system for DAC based on so-called building blocks, which are stable in form and size. Here, the level determines the possibilities for rearrangement: for level one, this is severely restricted, whilst for level four, it is highly flexible. Thus, this level system is a kind of transition from DAC to DAS.

2.2 Calculation of task load

Suitable task load models provide a fundamental background, which offers the opportunity to equally distribute the expected work load among air traffic controllers in adjacent sectors. As basis for the calculation of task load times, a comprehensive study was carried out at the DLR (Meinecke, 2014). Hence, a system of tasks and subtasks was defined and time values for the duration of each task and the frequency (if necessary) were advised. Further investigations could also include air traffic complexity, as driver for controller workload (Djokic et al., 2010).

The definitions, subdivision of controller tasks and times used by several companies, such as the German air navigation service provider DFS or EUROCONTROL, were taken into account. As a result, a system of 55 tasks for radar, planning, arrival, airport, tower, and apron controllers were defined with a set T_{ST} of 129 subtasks, grouped by Monitoring (MO), Radio Telephony (RT), Coordination (CO), Clearances (CL), Conflict Search (CS) and Conflict Resolution (CR). For every subtask t a task load time t_{tlt} was defined. Subsequently, all task load values for all aircraft inside the sector borders were summarised to a sector task load value TL_S .

Let A_S be the set of all aircraft moving within sector S , T_a the set of all task load subtasks from set T_{ST} applicable for aircraft $a \in A_S$, n_t the number of events of subtask $t \in T_{ST}$ and t_{tlt} the time needed to fulfil this subtask, then the task load TL_S for a sector S is defined as follows:

$$TL_S = \sum_{a \in A_S} \sum_{t \in T_a} n_t \cdot t_{tlt} \quad (1)$$

Table 1 exemplarily shows some tasks for a radar controller.

Main Type	Sub-Type	Task Name	Time (s)	per x Sec.	Group
Sector Entry	CHANGE_SECTOR_IN_CRUISE_FROM_SAME_ACC	Initial Call	11	-	RT
		Initial Monitoring	14	-	MO
		Receipt Flight Strip	3	-	CO
	CHANGE_SECTOR_IN_CRUISE_FROM_DIFF_ACC	Initial Call	15	-	RT
		Initial Monitoring	14	-	MO
		Receipt Flight Strip	3	-	CO
Conflict	CONFLICT_TYPE_1 (Consecutive flights, cruise)	Conflict Detection	17	-	CS
		Conflict Resolution	60	-	CR
Recurring Monitoring	RECURRING_MONITORING	Monitoring	5	120	MO

Table 1: Example of some classification types and task load times for controller actions within a radar sector.

2.3 Simulation framework

To provide a dynamic sectorisation, the calculation consists of three steps: (1) aggregation of traffic data, (2) initial sector structure and (3) optimisation. This section provides a general introduction of the fundamental principles and introduces the tool chain used to conduct the experiments described in this paper. For this, a simulation framework with three tools has been established (Gerdes, Temme and Schultz, 2016). The framework consists of the tools PrePro (pre-processor), RouGe (route generator), and AutoSec (optimisation framework), see Figure 1.

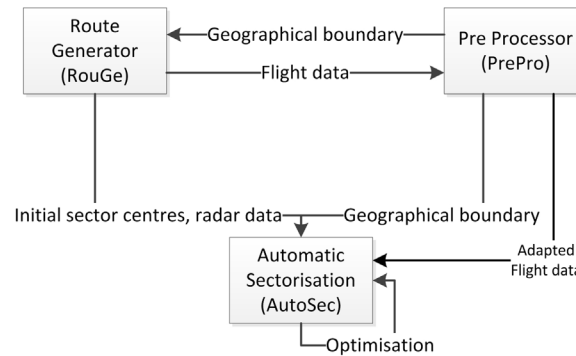


Figure 1: Generation of sector structure depending on task load.

RouGe and PrePro are developed and used to generate the necessary airspace data based on e.g. Eurocontrol's DDR2 data sets. Furthermore, the boundary of the airspace area of interest is defined and created using PrePro. This could be any connected area of airspace with one continuous boundary independent of national borders, fulfilling the connectivity constraint mentioned by several authors (cf. Flener and Pearson, 2013). PrePro extracts data of the real sectorisation, e.g. from DDR2 data, and feeds them into the optimisation tool AutoSec. RouGe uses the boundary created with PrePro to filter aircraft position data in dependence on parameters such as the given boundary, flight level and time. The resulting flight track data are then used to compute cluster centres using fuzzy clustering methods. These cluster centres are further used in AutoSec to construct a Voronoi diagram. Subsequently, the selected flights are sent to PrePro, which decreases the number of trajectory points for each flight.

The next step is carried out by AutoSec and depends on the type of experiment. AutoSec is able to create a Voronoi diagram based on a group of sector centres (e.g. cluster centres of RouGe) as the start sectorisation or to import a real sectorisation exported from PrePro. The start sectorisation is handed over to an evolutionary algorithm (sections 3.3, 4.3), which then optimises the selected sectorisation in dependence on a given evaluation function. The evolutionary algorithm shifts the sector boundaries in order to create an appropriate solution with respect to evaluation criteria such as task load and task load standard deviation (cf. Gerdes, Temme and Schultz, 2016).

For the optimisation process, a time component is included by defining time intervals and using different start sectorisations/cluster sets for each time interval. Thus, changing demands within a day's traffic flow, e.g. caused by changed weather conditions, are considered. The separate optimisation for successive time intervals can lead to considerable differences in sector structure, so the concept of interim diagrams was developed to smooth the transitions between consecutive sectorisations (Standfuß et al., 2018).

The sequence of steps for the creation of an optimised sectorisation is:

1. Creation of a boundary for a predefined airspace area (PrePro).
2. Creation of flight data sets for each predefined time interval of a day (RouGe).
3. Calculation of cluster centres for each flight data set using fuzzy clustering methods (RouGe).
4. Calculation of a convex Voronoi diagram as initial sectorisation based on the cluster centres for each time interval created in step 3 as initial solution for step 6 (AutoSec).
5. Application of a non-convex boundary form to the resulting Voronoi diagram of step 4 (AutoSec).
6. Optimisation of the resulting diagram by an evolutionary algorithm with respect to a certain optimisation task (e.g. uniform task load distribution, AutoSec).

Optional:

7. Subdivision of each time interval into an even number of sub-intervals and creation of interim diagrams for each sub-interval based on the surrounding optimised Voronoi diagrams (AutoSec).
8. Adaptation of the interim diagrams by an evolutionary algorithm with respect to a certain optimisation task and the structure of the underlying optimised Voronoi diagrams (AutoSec).

3 Methodological foundations

3.1 Fuzzy Clustering

In general, the aim of clustering techniques is to find a partition of a given dataset. In a fuzzy partition, a point is not necessarily assigned to a unique class or cluster (see Figure 2, Figure 3). Instead, membership degrees are associated with each point and each cluster. These membership degrees provide information about the ambiguity of the classification. Fuzzy clustering techniques are able to adapt to noisy data and to classes which are not well-separated. Here, we consider aircraft position data (trajectories) as data points and a clear separation of clusters might be difficult. The clustering solution consists of cluster centres, i.e. the centre of gravity of a cluster w.r.t. a distance function, and membership degrees. For an overview on fuzzy clustering and its applications, see de Oliveira and Pedrycz (2007) or Keller (2002). For our approach, we use a variant of the fuzzy-c-means clustering called probabilistic clustering (Bezdek, 1981) that is based on the great circle distance. In probabilistic clustering, the sum of membership degrees of a data point has to be equal to one and each membership degree has to be greater than zero.

Knowing the favoured number of sectors in advance leads to a fixed number of clusters that have to be determined. If the number of clusters is unknown, unsupervised clustering methods applying global validity measures can be used; cf. Keller (2002). The basic idea of unsupervised clustering is to define an upper (and eventually lower) bound of clusters and carry out the clustering for each number of clusters. A global validity measure is used to rate the partition as a whole. The ratings of different partitions are compared. Examining the curve of the validity measure over the number of clusters might show not only a best solution but also local extrema where a great gradient to neighbouring cluster numbers indicates reliable solutions. Suited to our problem are e.g. validity measures such as partition coefficient or partition entropy that rate the crispness of the partition, see Bezdek (1981).

Figure 2 demonstrates the clustering result for an example area for one time interval. The data points are associated to the cluster centre to which they have the highest membership degree, i.e. represented by the black lines.

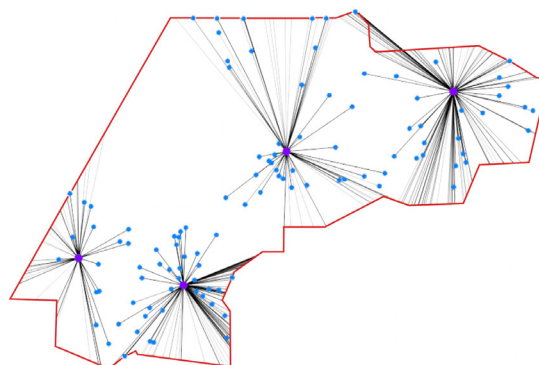


Figure 2: Cluster centre for time interval 1, EDYYDUTA.

3.2 Voronoi Diagrams

A Voronoi diagram (cf. de Berg et al., 2008; Xue, 2009) is a possibility for structuring a plane in dependence on a certain number of so-called centre points into sections where every point belongs to the section with the nearest centre point. Edges are created of all points which belong to two centre points, i.e. the points of an edge have the same distance to the two centre points, and vertices are those points which are associated with three different centre points (see Figure 3). An area containing all points with the same nearest point is called “face”. A convex hull is added once the faces have been determined. Often this convex hull simply has the form of a rectangle. With this definition, vertices are the backbone of a Voronoi diagram, because a shift in the vertex positions will rearrange the whole Voronoi diagram. Therefore, the vertices are the elements to be optimised by the evolutionary algorithm of AutoSec.

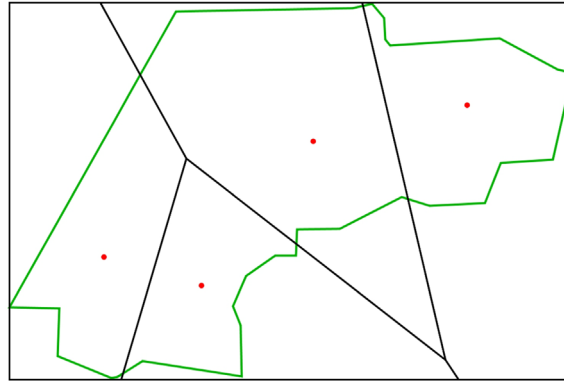


Figure 3: Voronoi diagram with rectangle as convex boundary (black), real boundary (green) and cluster centres (red).

3.3 Evolutionary Algorithms

The principles of evolutionary algorithms follow the evolutionary theory from biology (cf. Michalewicz, 1996; Gerdes et al., 2004) and are used in several aviation-related research fields, such as network planning (Kölker and Lütjens, 2015), airline crew pairing (Deveci and Demirel, 2018) and aircraft boarding (Soolaki et al., 2012; Schultz, 2017). An approach to apply genetic algorithms to dynamic airspace configuration is presented in Sergeeva, M. et. al. (2017). In nature, a group of individuals mix their genetic material, especially the information coded in chromosomes, to obtain a better chance of survival in a hostile environment through a higher degree of adaptation. For an evolutionary algorithm, a population with a predefined number of solutions for an artificial problem is created, where each solution is coded as sequence (chromosome) of parameters (genes) describing a possible problem solution. Typically, the chromosomes are of constant length. As in nature, solutions can be mixed between two chromosomes (so-called crossover), or some genes of a single chromosome can be mutated. To guarantee the “survival of the fittest”, an evaluation function *evalF* is created which rates each solution and supervises the selection of the fittest chromosomes for the next generation. These will undergo the evolutionary operator’s crossover and mutation again until an appropriate solution is found. This process is normally performed for a fixed number of generations or a prescribed distance of the evaluation value to a known solution.

3.4 Data structure

To store data of a Voronoi diagram, an appropriate data structure called Doubly Connected Edge List (DCEL, Berg et al., 2008) is used. A DCEL consists of three lists for vertices, half-edges and faces, where the elements of every list are connected in several ways. For the edge list, each (undirected) edge of the Voronoi diagram is divided into a pair of two half-edges (directed) with opposite directions called twins (see Figure 4), e.g. e_{11} / e_{11}^* . Furthermore, the information about the previous and the next half-edge when moving counter-clockwise through the half-edges of the corresponding face, the origin vertex, and a pointer to the corresponding face are stored with every half-edge additionally. Faces are finally defined by the centre point and a pointer to a single half-edge belonging to this face. So, every half-edge is linked to the corresponding face and each face is defined by only one corresponding half-edge. With this smart and appropriate structure, AutoSec is able to manage the framework of airspace within its software in an easy way, allowing the complete structure to be attained with a minimum of knowledge, e.g. moving through a complete sector when knowing only one edge and identifying all related sectors when knowing only one vertex. In particular with this data structure, it is possible to introduce the authentic, non-convex boundary forms presented in the next section.

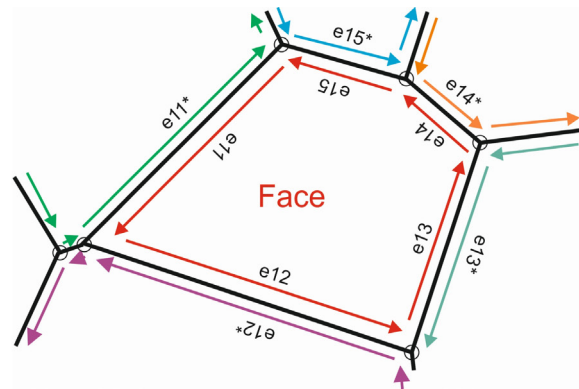


Figure 4: Example for the segmentation of a face into a system of half-edges and twins (marked with*). E11* is twin, e15 is the previous and e12 the next half-edge of e11.

4 Theoretical background and adaptations to the sectorisation problem

4.1 Non-convex boundaries

Another problem to be solved is the integration of a non-convex boundary line for an airspace region. Voronoi diagrams are limited by a convex hull. In the case of airspace areas which are taken into account to build the sectors, this cannot be guaranteed. Therefore, the calculation - carried out with a Fortune algorithm (Fortune, 1986) - has been extended in such a way that a Voronoi diagram can be adapted to arbitrary boundaries (Gerdes, Temme and Schultz, 2016). To start with, the Fortune algorithm is applied to the closest rectangle including the selected boundary (see Figure 3) which is clearly convex. The workflow of a sectorisation considering an authentic, non-convex boundary polygon is based on a “line-segment-intersection” method (Berg et al., 2008): The necessary steps are listed in the following.

1. Transform the boundary polygon into another DCEL.
2. Calculate coordinates for all breakpoints between the half-edges of both DCELs and sort the breakpoint list with increasing y-coordinate values.
3. Copy both DCEL's into a common DCEL (overlay).
4. Move through the breakpoint list and reconstruct the affected half-edges and vertices (create new half-edges and vertices, assign new previous, next and twin pointers) and create a new face list for the overlay DCEL (see Figure 5).
5. Remove all vertices, half-edges, and faces which are outside the boundary polygon.

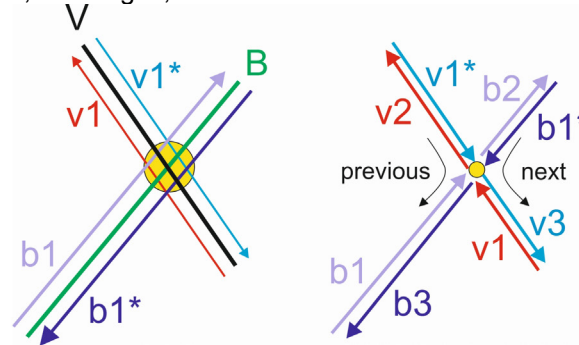


Figure 5: Cutting and rearrangement of half-edges in case of a breakpoint (yellow) and setting of previous and next pointers.

The resulting structure after step five is not necessarily a Voronoi diagram (see Figure 6). It will therefore be addressed as an initially adapted Voronoi diagram instead. In the case of a sector being divided and no centre point being included in a new sector, the centre of gravity is calculated as new centre point for each newly created sector.

4.2 Integration of a dynamic time-component

Due to different time zones in the world and the traffic between them, the structure of the air traffic does not remain constant throughout the day. This results in the necessity to define different airspace structures for the course of the day where each sectorisation is adapted to the current amount and distribution of air traffic. To increase DAS acceptance of airspace controllers, we aim at a smooth transition between successive sectorisations. Furthermore, smooth transitions between different sectorisations are operationally mandatory in order to prevent flights from leaving one sector, entering

the next and then jumping back to the first because new sectorisation demands it. For this transition, interim diagrams are introduced which are inserted between the sectorisations.

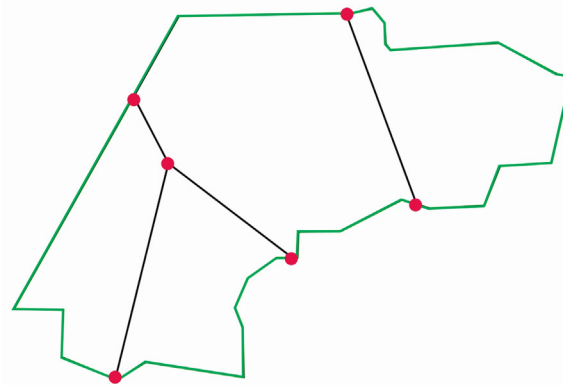


Figure 6: Example for a Voronoi diagram for EDYYDUTA (vertices as red dots, boundary in green).

The general course of action for the creation of interim diagrams is therefore as follows:

1. Calculation of Voronoi diagrams for each set of centre points.
2. Mapping of vertices between successive Voronoi diagrams as far as possible.
3. Splitting of the time intervals into smaller sub-intervals.
4. Application of evolutionary algorithm with normal evaluation function to each Voronoi diagram to generate a so-called optimised Voronoi diagram (i.e. not necessarily a Voronoi diagram).
5. Calculation of the corresponding number of interim diagrams (not necessarily Voronoi diagrams) taking the mapped vertices of the optimised Voronoi diagrams into account.
6. Application of the evolutionary algorithm.

The structure of Voronoi diagrams depends on the number and position of the centre points. When using time-dependent sets of centre points, the resulting Voronoi diagrams can be very different in the number of faces as well as in position and number of vertices. The target is to insert interim diagrams between successive pairs of Voronoi diagrams which should reflect the structure of both involved Voronoi diagrams. Therefore, it is necessary to identify those faces and vertices which exist in both diagrams, because they can be used to calculate interim vertices between the original vertices. This is done by using information about the position of the centre points, the position of the vertices and the location of the sector itself (inner/outer sector).

The mapping algorithm starts by identifying all faces next to the boundary for both Voronoi diagrams. Subsequently, for every face of the Voronoi diagram with the lower number of faces, a mapping face from the other Voronoi diagram is searched by calculating the distances between the centre points and distances between the vertices and selecting the nearest centre point with the same face type (inner/outer). In the next step, all vertices are mapped which are (nearly) the same (e.g. boundary vertices) or start at the boundary. For the latter, the adjacent faces have to be known (Figure 7). For the last step, all faces are ordered in a sequence depending on the relation of the number of already mapped to all vertices of this face. Then, the last unmapped vertices for all faces are subsequently mapped in the same way as the faces before by using distance and type of vertex (inner/outer/starting at boundary).



Figure 7: Identification of mapped boundary vertices V_1 and V'_1 using mapped faces.

This method works from outside to inside by identifying the outer sectors first and then using the information to identify mappings for the adjacent sectors. This procedure has been tested for parts of upper airspace divided into a maximum number of eight sectors so far and worked well. Nevertheless, with an increasing number of sectors, this procedure will be subject to further investigations and improvements.

The connected vertex in a successive Voronoi diagram is called Map, in a preceding diagram preMap. If successive Voronoi diagrams have different vertex numbers, as many connected vertices as possible are identified. The remaining vertices stay unchanged for the interim diagrams.

As previously mentioned, the most difficult challenge for the creation of interim diagrams arises from the fact that surrounding optimised Voronoi diagrams possess significantly different structures. To cope with this, the number (nr) of interim diagrams inserted between two optimised Voronoi diagrams should be even-numbered, so that it can be divided into two equally numbered groups. The first group, consisting of $\{V_1, \dots, V_{nr/2}\}$, then receives structure and data from the first optimised Voronoi diagram V_0 , whilst the second group, consisting of $\{V_{nr/2+1}, \dots, V_{nr}\}$, receives structure and data from the second optimised Voronoi diagram V_{nr+1} . In the next step, the coordinates of all mapped vertices are recalculated for the interim diagrams proportionate to the distance between the vertex of the first optimised Voronoi diagram and its mapping.

For a given sequence of diagrams $V_0, V_1, \dots, V_n, V_{n+1}$, where V_0 and V_{n+1} are optimised Voronoi diagrams and V_1, \dots, V_n the interim diagrams between them, the positions for those vertices of the interim diagrams where a mapping exists are calculated with (2): let $V_i[j]$ be the vertex j of diagram V_i , $i \in \{1, \dots, nr\}$, $V_0^{map}[j] = V_{nr+1}[k]$ the vertex k of the second optimised Voronoi diagram - to which vertex j of the first optimised Voronoi diagram is mapped.

$$\begin{aligned} V_i[j](x) &= V_0[j](x) + \left(\frac{i}{nr+1}\right) * (V_{nr+1}[k](x) - V_0[j](x)) \\ V_i[j](y) &= V_0[j](y) + \left(\frac{i}{nr+1}\right) * (V_{nr+1}[k](y) - V_0[j](y)) \end{aligned}, \quad i \in \left\{1, \dots, \frac{nr}{2}\right\} \quad (2)$$

A similar formula based on the data of the second Voronoi diagram is used for the calculation of the coordinates of the second group of interim diagrams V_i , $i \in \{nr/2 + 1, \dots, nr\}$. Vertices without a mapped vertex remain unchanged. Figure 8 shows a graphical interpretation of this procedure. Here, parts of a DCEL for the first (blue) and the second (yellow) optimised Voronoi diagrams are shown. In the next step, two new diagrams are calculated, where the structure of the first one relies on the first optimised Voronoi diagram and the second one on the structure of the second. Then the coordinates are adapted in such a way that they consist of 2/3 (bigger arrow) of the first and 1/3 (smaller arrow) of the second diagram for the first interim diagram and vice versa for the second. So, $I4$ moves upwards and $I4^*$ downwards in comparison to the original vertex positions, and $V5^*$ remains unchanged.

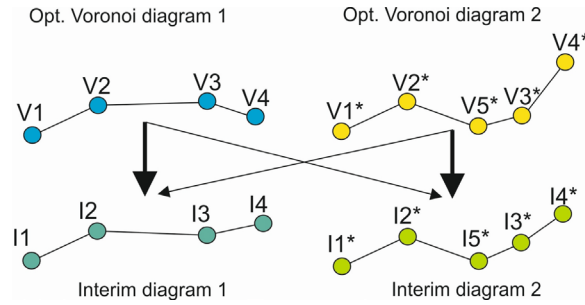


Figure 8: Example for the creation of two interim diagrams between a set of vertices of two Voronoi diagrams.

4.3 Evolutionary Algorithm of AutoSec

Within this section, the used evolutionary algorithm with the problem specific adaptation is described, consisting of the formal description of the search space, description of the evolutionary operator's mutation and crossover, the evaluation function and selection process.

For the evolutionary algorithm, a rectangular portion AS of airspace was selected with $AS = \{(x, y) | x_u \leq x \leq x_l, y_l \leq y \leq y_u\}$, where $u = (x_u, y_u) \in \mathbb{R}^2$ denotes the upper left and $l = (x_l, y_l) \in \mathbb{R}^2$ the lower right corner. Let B be the boundary described as polygon $B = (b_1, \dots, b_m)$ of length m with $b_i \in AS, i \in \{1, \dots, m\}$, n the proposed number of sectors and k the number of vertices of the resulting Voronoi diagram. Then

$$AS_B = \{(x, y) | (x, y) \in AS \wedge (x, y) \text{ lays on boundary } B\} \quad (3)$$

is the set of all points on the boundary B and

$$AS_{IB} = \{(x, y) | (x, y) \in AS \wedge (x, y) \text{ inside boundary } B\} \quad (4)$$

the set of all points inside B .

The vertices $v_i \in AS, i \in \{1, \dots, k\}$ of the original Voronoi diagram (section 3.2) are used directly as genes and a sequence of these vertices with a predefined order forms a chromosome $c = (v_1, \dots, v_k)$ (Gerdes, Klawonn and Kruse, 2004). Therefore, each chromosome represents a complete sectorisation determined by the structure of the Voronoi diagram calculated beforehand. The population contains in each chromosome a valid list of vertices.

The general structure of the selected evolutionary algorithm is based on the modGA introduced by Michalewicz, Z. (1996). It is especially designed to prevent so-called super-individuals from overtaking the population. For this, three groups of chromosomes are created for each generation, where the chromosomes of the first group are all different and remain unchanged. They are selected using a stochastic sampling selection (cf. Michalewicz, Z., 1996), but without replacement. The elements for the other two groups are selected using stochastic sampling with replacement. The chromosomes of the second group undergo the crossover operator, whilst the chromosomes of the third group are mutated. So, crossover is responsible for composing new solutions from existing ones and the mutation operator carries out small steps to improve the solution.

As crossover operator for a selected pair of parent chromosomes, the well-known uniform crossover was used (c.f. Michalewicz, Z., 1996), where every gene of the first chromosome has the same prescribed chance of being exchanged with the corresponding vertex of the other parent chromosome. Within the chromosomes, the position of a vertex is not important; it is therefore not necessary to exchange parts of the chromosomes as it is done using one or two-point crossover.

When applying the mutation operator, it has to be guaranteed that a vertex $v \in AS_B$ stays at the boundary after mutation and a mutated inner vertex $v \in AS_{IB}$ stays inside the boundary. This problem was solved by introducing two adaptations to the algorithm structure and especially to the mutation operator. The first adaptation is the replacement of the part of the boundary polygon of the associated face/sector by the two boundary breakpoints (see Figure 9). These breakpoints are added as additional artificial vertices to the adapted Voronoi diagram and are mutated only by moving them on the boundary. Now, the number of vertices for each face always stays the same and a mutation operator is defined for every type of vertex (inner/boundary).

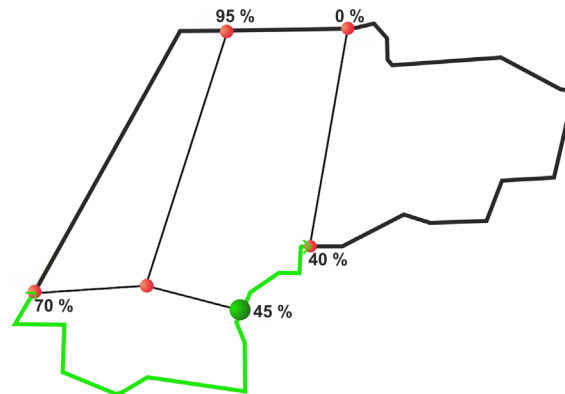


Figure 9: Optimised Voronoi diagram with vertices as red dots, exemplarily showing a mutation between 40 and 70% for green vertex (45%).

For the second adaptation, the boundary is considered as a straight line and a percent value is allocated to the relative position of each outer vertex on this line (breakpoint between faces and boundary polygon). With this adaptation, the mutation of a boundary vertex is handled as the random selection of a percentage between the percentage values of the surrounding boundary vertices. Figure 9 shows the mutation area (green line between 40 and 70%) for the outer vertex at 45%.

For the use of interim diagrams, a special equation to limit the mutation of vertex coordinates was created. In the case of equal Map and preMap, the difference between the new and the original vertex is calculated and set in relation to a predefined threshold value, where values above the threshold are penalised by doubling the distance. In the case of different Map and preMap, an ellipsoid between Map and preMap is used to define an allowed area for the location of a new vertex point (see Figure 10). The border of this area is defined by all points where the sum of the distances of the point to the focus points F_1 and F_2 is the same. Each point outside this ellipsoid is penalised. The difference to the original vertex is calculated for each vertex and called *vertexCloseness* of an interim vertex. In the case of unmapped vertices, a predefined mutation limit is used.

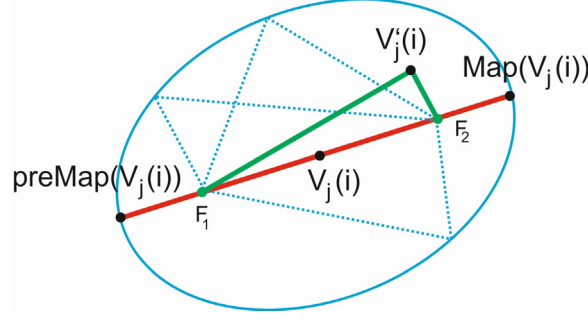


Figure 10: Definition of an allowed area for vertex $V_j(i)$ with different Map and preMap. Blue dotted lines show examples for border points, green line a possible new position $V_j'(i)$ inside the allowed area.

We have tested several variations of factors for the evaluation function of the evolutionary algorithm (see Gerdes, Temme and Schultz, 2016). From these, the following five elements were selected for the simulations presented here:

- Sum of task load over all sectors (TL).
- Standard deviation of task load between sectors (TLSLSD).
- Standard deviation of interior angles (A).
- Number of flight intervals (partition of flights by sectors) over all sectors (FI).
- Closeness of vertices (VC, optional, in case interim diagrams are in use).

With this selection of evaluation parameters, the problem can be seen as a multi-objective optimisation problem, as stated by Zou et al. (2016). Whilst the first two elements are directly connected to the task load distribution problem (balanced work flow for airspace controllers), the third and fourth are used to ensure an operationally relevant (more “usable”) sector layout. The calculation of task load was described in section 2.2. and the interior angles of a structure are the angles between successive edges. So, the standard deviation of interior angles measures the difference to a polygon with uniform angles. Delahaye, Schoenauer and Alliot (1998) and Kulkarni, Ganesan and Sherry (2011) have stated that a convexity constraint should be applied to created sectors to ensure that an aircraft visits every sector only once (Flener and Pearson, 2013). They also demanded a constraint limiting the inter-sector flow. These constraints are included indirectly by the evaluation factors “standard deviation of interior angle” and “number of flight intervals” but not as mandatory restriction. Including the number of flights per sector indirectly results in avoidance of short dwell times of flights in sectors and represents segmentation of flights over different sectors. As far as the convexity constraint is concerned, today’s sectors are not necessarily convex. Due to the form of the evaluation function, Pareto optimal results with respect to task load and task load standard deviation can be expected, where the shape may look completely different. The optimisation is based on a fixed flight plan, which remains unchanged for the optimisation process. Since no traffic movement simulation takes place, delay is not evaluated.

To avoid the influence of incomparable value ranges for the different factors of the evaluation function, we developed a combined function using a ranking approach where the influence factors are weighted (Standfuß et al., 2018). These weights are not necessarily constant during the simulation run, but are adapted to some extent during optimisation. This is the case for w_{TLSLSD} because TL and $TLSLSD$ act in opposite directions and have to be optimised consecutively (Gerdes et al., 2016.).

$$evalF = w_{TL} \cdot E_{TL} + w_{TLSLSD} \cdot E_{TLSLSD} + w_A \cdot E_A + w_{FI} \cdot E_{FI} + w_{VC} \cdot E_{VC} \quad (5)$$

$$\text{with } w_{TL} = 1, w_A = 0.5, w_F = 0.5,$$

$$w_{TLSLSD} = (1 + w_{TL} + w_A + w_{FI}) - \left(\frac{gen_{nr} \cdot 1.5}{gen_{max}} \right)$$

In (5), gen_{nr} is the number of the actual generation, gen_{max} is the maximum number of generations to be calculated and w_i are the weights for the corresponding evaluation factors E_i , $i \in \{TL, TLSLSD, A, FI, VC\}$. With an evaluation function consisting of several - partly contrary - elements, it is possible to create sectorisations which differ widely in shape but show a very similar evaluation value. Furthermore, when an optimisation is carried out where interim diagrams are created, an additional part $w_{VC} \cdot E_{VC}$ is added to the evaluation function representing the vertex closeness.

5 Experimental setup

As airspace region, EDYYDUTA (EDYY Deco Sectors, Maastricht) was selected as demonstrative example because of its clear structure with only four sectors. Furthermore, this airspace region was

also used by Sergeeva et al. (2015) for several simulations for their DAC approach and allows a comparison between DAC and our DAS approach. To take a sufficient number of flights as basis, the flight level range was set to the region from flight level 300 to unbounded (standard level for higher altitude is 345 for this airspace region). The number of cluster centres is equal to the number of real airspace sectors.

For the experiments presented in this paper, the air traffic movements are extracted from Eurocontrol DDR2 data set for 12.07.2012 by RouGe and are divided into four time intervals given in hours of a day: $[0,4]$, $[4,12]$, $[12,20]$ and $[20,24]$ with centres at 0, 8, 16 and 24. The first and the last intervals are smaller because with 0, resp. 24, as centre they are extended to the time before and after the selected day (see Figure 11, above), but only the traffic of the selected day is taken into account. When using interim diagrams, these time intervals are shortened for the main Voronoi diagrams and the selected number of interim diagrams is inserted (see Figure 11, below).

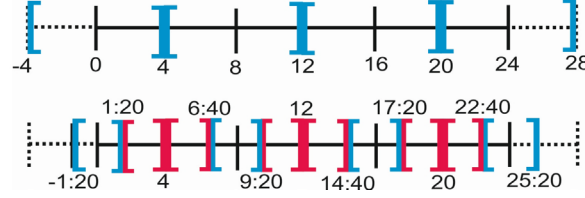


Figure 11: Example for the creation of time intervals (hours of a day) for main intervals (above) and when using two interim diagrams (below). Main intervals are blue, interim diagrams are red.

As simulation setup for the evolutionary algorithm, a population size of 80 chromosomes was used and the number of generations to be simulated was set to 1000. Ten simulations were carried out for each sectorisation. The flights selected by RouGe are defined as a sequence of connected points $p \in AS_B \cup AS_{IB}$ (see (3), (4)). Tests for an appropriate set of simulation parameters (e.g. crossover and mutation rates) were presented by Gerdes, Temme and Schultz (2016) and result in a mutation probability of 5% and a crossover probability of 20%. The crossover probability can be much higher than for the mutation, because the exchanged section of a chromosome often has mainly the same genes (vertices) and therefore a substantial change of the solution takes place less often.

The evaluation value for the closeness of a chromosome consists of the sum of the *vertexCloseness* for all vertices. In addition, the weight factor w_{vc} for the new part E_{vc} of the evaluation function was investigated and set to three, when interim diagrams are in use (Gerdes, Temme and Schultz, 2016). Because the *vertexCloseness* is calculated as a penalty and not as criterion for exclusion, it is possible that the interim diagrams differ recognisably from the surrounding main diagrams if this is justified by better values for task load and task load standard deviation.

6 Results

For a better understanding of the abilities and possibilities given by our tool chain (cf. section 2.3), we have selected four scenarios for comparison: the real sectorisation and its optimisation as well as an artificial Voronoi structure and its optimisation. The results are presented and compared in this section. Furthermore, the Voronoi sectorisation and its optimised version are used for the creation of interim diagrams with two inserted diagrams for each of the four main intervals. The results are analysed in regard to the overlapping airspace areas, respectively building blocks.

To obtain an impression of the differences between the calculated cluster centres for the Voronoi-based sectorisations, the mean cluster centres over all time intervals were calculated and compared to the cluster centres for each time interval (

Table 2). For this comparison, the centres were associated to the closest centre mean. The highest difference was found for time interval 0. Nevertheless, for the historical flight data set, the traffic for the different time intervals was very similar, especially for time intervals one and two. They have the highest amount of traffic and therefore the centres are more stable than for intervals with lower traffic numbers.

Centre	Mean		Distances in Time Interval			
	x	y	0	1	2	3
1	284.3	55.2	9.2	4.7	4.1	8.4
2	114.6	172.3	2.6	4.7	3.3	2.6
3	45.8	160.6	13.4	2.5	4.2	8.7
4	180.8	87.8	5.3	5.2	5.6	5.3

Table 2: Position difference (distance) between centres for the selected time intervals (NM)

6.1 Optimisation of real sectorisation of EDYYDUTA

For this part of the experiment, a real sectorisation (baseline real sectorisation: **bRS**) of the upper airspace is used. Because optimising a real structure leads to a new configuration, this approach belongs to the DAC approaches. The selected airspace area EDYYDUTA consists of four sectors (airspace blocks) with a resulting number of 12 vertices for the inner structure (Figure 12, black lines). Only these vertices undergo the optimisation process using the developed evolutionary algorithm (see section 4.3), which always starts with the same set of vertices for the different time intervals. Because one focus is set on the sequence of sectorisations for successive time intervals, the positions of these vertices are major parameters for similarity in the geometric shape of the sectors.

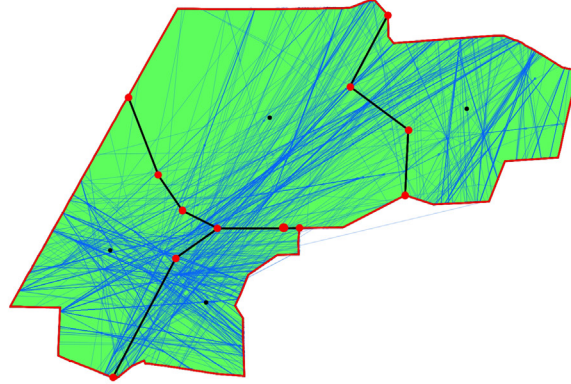


Figure 12: EDYYDUTA original sectorisation with centre of gravity for each sector (**bRS**). Air traffic is shown with blue lines; darker colour implies higher number of flights. Airspace sectors are separated with black lines.

Since the optimisation of airspace sectors could result in different geometric shapes, but with same values for the evaluation function, the average position for each vertex was exemplarily calculated in ten simulation runs for the optimised real sectorisation (**oRS**). A typical result for one simulation run is shown in Figure 13. The vertices are numbered and marked by red dots; the vertices 1, 4, 5, 9 and 12 are located at the boundary.

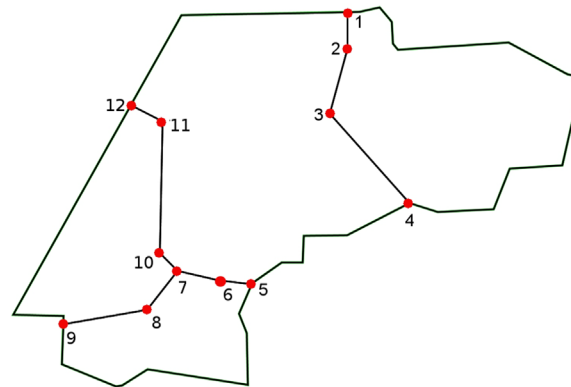
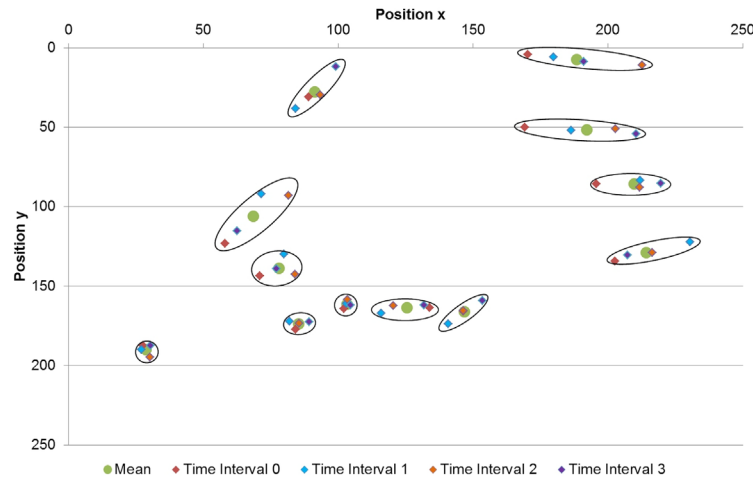


Figure 13: Example for **oRS** for time interval 2 (12:00 – 20:00) with vertices as red dots.

Figure 14 provides a more detailed analysis of the vertex positions. Here, the mean values for the vertices are shown as green circles, whilst the average values for the four time intervals are marked by rhombi. The comparison of Figure 12 (baseline layout) with Figure 14 (variation of optimised layouts) shows that the border between centre and right sector, formed by vertices one to four, is much straighter for **oRS** than for the baseline version **bRS** of EDYYDUTA (except time interval 0) when considering the mean values.

Figure 14: Position difference between vertices for the selected time intervals for **oRS**.

Nevertheless, the ellipsoids around each group of vertices showing the variability of solutions include possible vertex positions similar to the vertex positions of **oRS**. The same can be recognised for vertices eleven and twelve. Contrary to this, the positions of the vertices five to ten are very limited in their position variability and have a completely different position in comparison to the **bRS**. All of them moved to the southern part of EDYYDUTA, resulting in smaller sectors and automatically in a reduction of task load. The task load values for the different task load types (excluding clearances, which are not given in this sector type) are presented in

Table 3 and

Table 4 for **bRS** and **oRS** (section 2.2).

Time Interval	Monitoring	Radio Telephony	Coordination	Conflict Search	Conflict Resolution
0	22727	7585	5669	1349	2920
1	44020	14767	11428	1642	2287
2	48200	15768	12081	1252	1095
3	22068	7222	5593	1293	2878

Table 3: Task Load Average Values **bRS** (seconds).

Time Interval	Monitoring	Radio Telephony	Coordination	Conflict Search	Conflict Resolution
0	22904	7575	5638	1325	2927
1	44057	14478	11220	1648	2340
2	47376	15445	11831	1186	1023
3	21741	7012	5422	1223	2757

Table 4: Task Load Average Values **oRS** (seconds).

The comparison of these values shows that they are very close together, but the **oRS** results are slightly better for most factors. The highest difference exists for monitoring for time interval two. Monitoring is carried out for sector entries on the one hand and as recurring monitoring every two minutes on the other. Therefore, a considerable reduction in these values is based on a reduced dissection of flights. A comparison of the factors of the evaluation function is presented in

Table 5. This shows indeed that the number of flight intervals is lower for the **oRS** version and here especially for time interval two. A closer look at the main factor task load standard deviation reveals furthermore that there is a very high difference between both versions and that the optimisation has mainly reduced this value whilst stabilising the task load on a comparable value. The task load standard deviation was reduced to values between 0.5% for time interval three and 6.9% for time interval two. Only the result for the factor task load in time interval zero is better for **bRS** than for **oRS**. It is obvious that the baseline structure with twelve vertices has given AutoSec the possibility of an easier adaptation of the distribution of task load to the different sectors than for reducing the task load as well. This is not at least due to the expert knowledge already included in the baseline structure of EDYYDUTA. The values for the interior angle are only slightly better than the baseline results. This proves the similarity of sector forms for both versions, even if the sectors themselves have different sizes.

Time Interval	Task Load Sum [seconds]		Task Load Std. Deviation [seconds]		Interior Angle Std. Deviation [°]		Flight Intervals [#]		Std. Deviation Area [NM]	
	<i>bRS</i>	<i>oRS</i>	<i>bRS</i>	<i>oRS</i>	<i>bRSe</i>	<i>oRS</i>	<i>bRS</i>	<i>oRS</i>	<i>bRS</i>	<i>oRS</i>
0	40251	40370	3171	43.2	45.8	45.2	389.0	386.2	5621	6074
1	74145	73745	4841	111.0	45.8	44.5	762.0	749.1	5621	6178
2	78397	76863	3755	260.5	45.8	44.6	802.0	785.0	5621	5833
3	39055	38156	2865	13.8	45.8	45.0	384.0	373.0	5621	6453

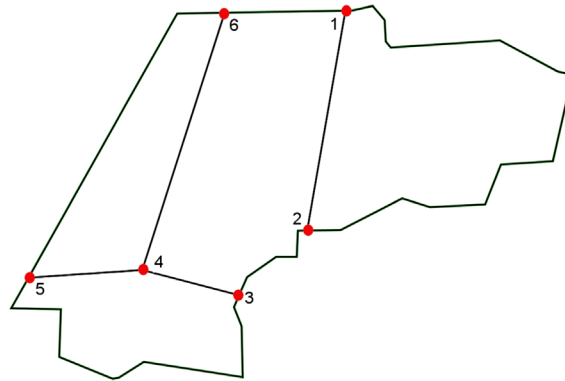
Table 5: Comparison of **bRS** and **oRS** for the factors of the evaluation function.

The factor “Std. Deviation Area” was not part of the evaluation function but is included in

Table 5 for a better impression of the re-arrangement of sector sizes. The reduction in task load standard deviation is accompanied by an increase in the difference of sector sizes. Altogether, the results prove that AutoSec is able to improve a sectorisation whilst staying close to the original sectorisation, which is a satisfactory initial result. Thus, it would be possible to re-use existing sectorisations, e.g. from a set of typical sectorisations calculated before, as start solution for a new traffic situation without changing the structure too much. The additional workload for controllers for the transition period can be kept low. In a future air traffic scenario, such as avoidance of contrails, this means that it is possible to create a set of typical sectorisations, select the most suitable and apply AutoSec to provide an appropriate airspace structure. This would reduce the effort for the creation of sectorisations and ensure some similarities among successive sectorisations.

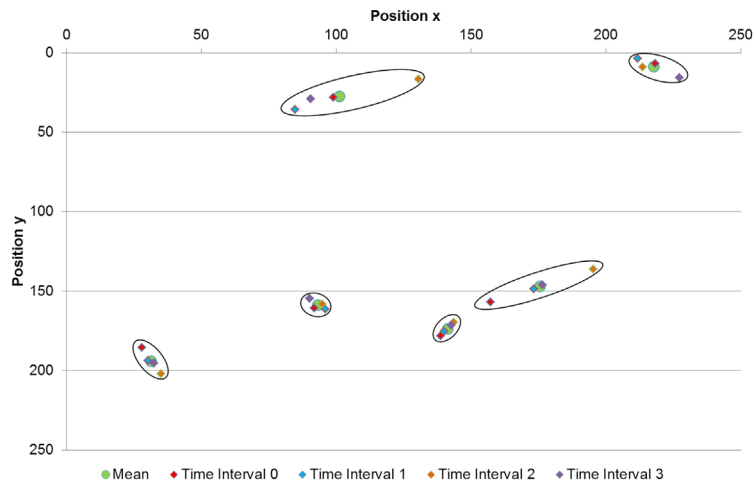
6.2 Optimisation of EDYYDUTA using dynamic airspace sectorisation

In contrast to **oRS**, where the optimisation for each time interval started with the same baseline structure, the structure for optimisation based on traffic cluster sectorisation (**oCS**) differs depending on the set of cluster points calculated by RouGe for each time interval. Therefore, this approach belongs to the DAS group. Nevertheless, for this example, the number of vertices for each time interval was the same. Figure 15 shows an example for an optimised sector structure for time interval two with six vertices.

Figure 15: Example for **oCS** for time interval 2 (12:00 – 20:00) with vertices as red dots.

The reduced number of vertices in comparison to **oRS** leads to a lower number of possibilities for sector forms. Therefore, it is much more difficult to justify the vertex positions in order to obtain good results for all parts of the evaluation function. In Figure 16, the average vertex positions for each time interval and the mean overall time intervals are presented in the same way as for DAC in Figure 14. Many of the vertices are very close together, but the results for vertices one, two and six show variations.

A closer look at the simulation result has shown that there are two equivalent solution types. Only the angle of the line between vertices one and two is different. The position of this line influences these three points and results in a very different distribution of task load and task load standard deviation for different optimisations. Nevertheless, the resulting sectorisations for the different time intervals are very similar in structure (see related ellipsoids in Figure 16). Therefore, no problems for controllers responsible for EDYYDUTA should occur when switching from one sectorisation to the next.

Figure 16: Position difference between vertices for the selected time intervals for **oCS**.

A closer look at the task load results presented in

Table 6 and

Table 7 reveals a considerable difference between baseline of traffic cluster based sectorisation (**bCS**) and optimised sectorisation (**oCS**). Only the value of conflict resolution for time interval 1 is higher in **oCS** than the baseline result.

This suggests that the evolutionary algorithm uses conflicts to balance the task load between sectors by placing a borderline close to a conflict and counting this conflict for each adjacent sector. To avoid this, the introduction of an additional parameter evaluating the number of conflicts for each sector or measuring the distance to the closest border will be part of further investigations.

Time Interval	Monitoring	Radio Telephony	Coordination	Conflict Search	Conflict Resolution
0	23530	8114	5987	1464	3186
1	45261	16366	12467	1686	2082
2	49497	17495	13216	1345	1030
3	23326	8457	6412	1332	2813

Table 6: Task Load Average Values **bCS**.

Time Interval	Monitoring	Radio Telephony	Coordination	Conflict Search	Conflict Resolution
0	22434	7289	5376	1291	2842
1	43798	14395	11023	1630	2313
2	47457	15459	11662	1163	963
3	21915	7155	5472	1253	2804

Table 7: Task Load Average Values **oCS**.

The results for the factors of the evaluation function can be found in

Table 8. The optimised results of the DAS approach for these factors are always better than the baseline results. For task load, they differ between approximately 3000 and 6000 seconds, and the task load standard deviation is reduced by more than 85% for each time interval. The optimisation functionality was able to reduce the main factors of the evaluation function significantly in spite of the low number of vertices and therefore borderlines. The dissection of flights was considerably reduced as well. This is another reason for the task load reduction, because the number of sector entries with its connected task load decreased. The highest value for the task load standard deviation for DAS can be found in time interval 2, together with the lowermost value for the area standard deviation, which is lower than the baseline value.

Time Interval	Task Load [seconds]		Task Load Std. Deviation [seconds]		Interior Angle Std. Deviation [°]		Flight Intervals [#]		Std. Deviation Area [NM]	
	bCS	oCS	bCS	oCS	bCS	oCS	bCS	oCS	bCS	oCS
0	42282	39235	3293	284	46.6	44.2	410.0	368.5	4014	4846
1	77862	73161	5629	344	46.9	44.4	829.0	736.4	4594	5170
2	82583	76706	5193	690	47.7	43.9	875.0	774.0	4457	4254
3	42340	38600	2923	438	46.6	44.9	437.0	376.5	4792	5443

Table 8: Comparison of **bCS** and **oCS** for the factors of the evaluation function.

6.3 Comparison and evaluation of simulation results

Within this section, a comparison of the results of optimised real and cluster-based sectorisation (**oRS**, **oCS**) is presented against corresponding baseline (**bRS**, **bCS**). The main difference between **bRS** and **bCS** lies in the higher number of vertices for DAC and the resulting lower flexibility for **bCS**. On the one hand, this leads to increased possibilities for creating different solutions with similar evaluation values for **oRS**, but a higher number of vertices can increase the search time and space. Since the **bRS** was created using expert knowledge, the larger search space can be compensated through a better start sectorisation and the results are much better than for **bCS**. Comparing the results for **bCS** in

Table 8 with the results for the **bRS** in

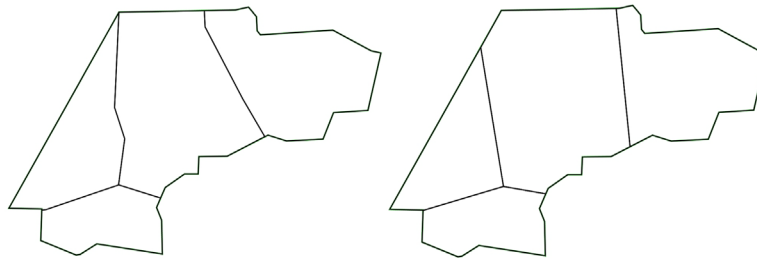
Table 5 shows immediately that **bRS** outperforms the DAS baseline for each factor and time interval, with the exception of the non-optimised standard deviation of the area. However, the results are close together, and this shows that even the optimised Voronoi diagram created for our DAS approach (which is **bCS**) can be used as a substitute for situations where no sectorisation created by experts exists or there is no time to create one by hand.

Because of the reduced number of vertices, the **oCS** sectorisation has a simpler structure than **oRS**. This is an advantage for situations where the sectorisation changes over time and controllers have to adapt their work procedures accordingly. Nevertheless, when comparing both elements of Figure 17, it is easy to see that the rough structure is similar to but less sophisticated than **oRS**.

When comparing Figure 14 and Figure 16, it is obvious that **oCS** tends towards a more stable distribution of vertex positions over the time intervals than **oRS**, caused by the higher number of vertices. For the task load types, the results are mixed and very close together for **oRS** and **oCS**, whilst both are better than **bCS**. For time interval three, task load values (monitoring, radio telephony, etc.) are higher for **oCS** than for **oRS**, resulting in a higher general task load value for time interval three. All other task load values are lower than the results for **oRS** (

Table 5,

Table 8), but all values for the task load standard deviation are much higher than for **oRS**. So, the decrease of the task load is paid for with an increase in task load deviation as could be expected from the evaluation function (see Eq. (4)), where both factors are in conflict.

Figure 17: Example for **oRS** (left) and **oCS** (right) for time interval 1.

The optimisation process for **oRS** has used the better starting condition to focus on task load standard deviation instead of task load as is performed by **oCS**. Because the task load is strongly influenced by the form of the sectors and consequentially by the number of flight intervals, these results are also mostly better than the related **oRS** values.

6.4 Creation of building blocks

An important point for the applicability of a new concept is the amount of workload additionally created for the users of the system, e.g. the airspace controllers. Especially in the case of dynamic sectorisation, the extent of workload caused by the change in shape and position of the artificial sectors

will influence the acceptance rate of this concept by airspace controllers. How the application of interim diagrams can help to cope with this problem is presented in this section.

Therefore, it is necessary to analyse the dissimilarity of subsequent sectorisations for a selected part of the airspace. Very important for such an analysis is the part of a sector which is the same for all subsequent time intervals. This main part can be seen as Sector Building Blocks (SBB), as described by Sergeeva et al. (2015). They describe the remaining airspace areas as Sharable Airspace Modules (SAM), because these parts of the airspace are shared amongst different sectors over the time. The idea of SBBs and SAMs will be applied to our DAS approach. Again, the EDYYDUTA airspace area is used with an **oCS** approach and the same parameters as for the preceding optimisation runs. To calculate these SBBs for EDYYDUTA, overlays for the sector shapes for each sector for all time intervals were created and the parts of the airspace shared by all time intervals were calculated. The result is used as SBB for each sector.

Figure 18 illustrates the results of an example optimisation for all sectors. The areas coloured with brighter shades of blue indicate that they are not occupied within all time intervals. The darker a colour is, the more sectors share this part of the airspace. The darkest areas are shared by all time intervals and can be seen as the SBBs mentioned before. Sectors zero and one have small overlapping areas, especially because both adjoin each other on the right, respectively left side.

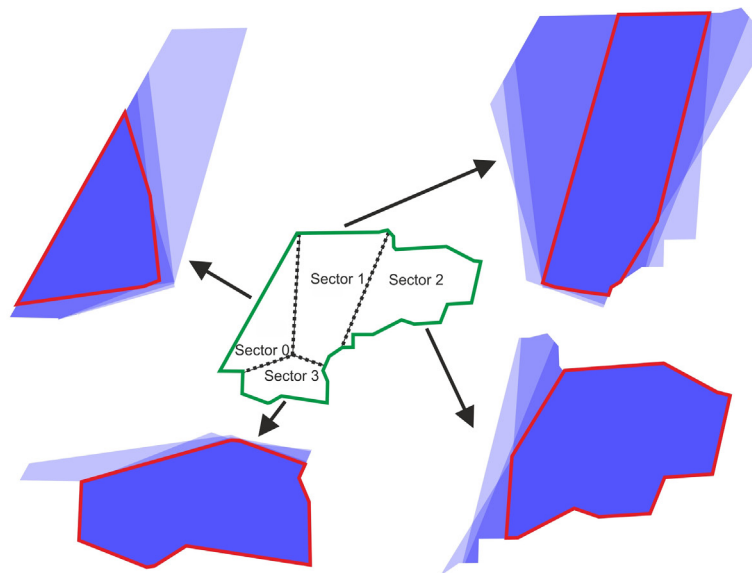


Figure 18: Example for the overlapping sectors (blue areas, the darker, the more sectors overlap) and the SBBs (bordered by red lines).

The overlapping areas to the south with sector three are obviously smaller than between zero and one. The SBBs for sectors two and three include more than 75% each of the area covered by all time intervals, which is a very good value. The percent values for the SBBs for all sectors are given in

Table 9.

	Sector 0	Sector 1	Sector 2	Sector 3
SBB (%)	44.4	48.2	75.3	78.8

Table 9: Example of size of Sector Building Blocks in percent.

The area of sector zero coloured with the lightest blue is undesirably large for the transition from one time step to the next, whilst all other SAMs are of a sufficient size to be switched on or off in dependence on the time interval. The transition from SBB to a combination with this large SAM could provoke situations where an aircraft has just left a sector shortly before a sectorisation change and is sent back to the same controller after it. To avoid these situations, interim diagrams can be used. The transition to the addition of a large type of SAM is divided into more steps of smaller size. So, an aircraft moving through this large SAM would hopefully stay inside the subsequently added smaller SAMs. To demonstrate the application of interim diagrams, additional simulations were carried out. For these, the same sector (EDYYDUTA) was used together with the same flight schedule and simulation parameters, but each time interval was divided into three additional time intervals by adding two more interim diagrams (see Figure 11).

An example for the usage of interim diagrams is shown in Figure 19 for sector zero. On the left, the main time intervals [0:00-1:20], [6:40-9:20], [14:40-17:20] and [22:40-24:00] are shown (Main SBB). All diagrams including the newly created interim diagrams are on the right side (All SBB). It can be seen that there is a large increase in sector area between time intervals zero and one for the main intervals, which is successfully divided into smaller parts when using the interim diagrams.

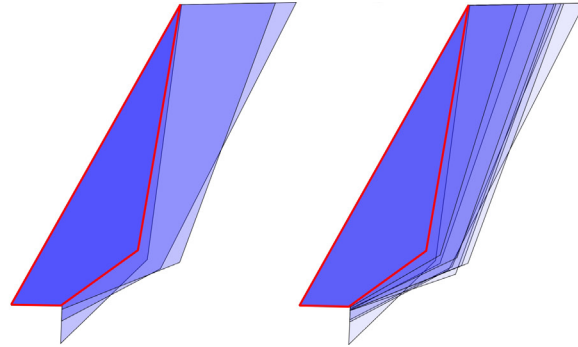


Figure 19: Example for application of interim diagrams for sector one: Main diagrams (left) and including interim diagrams (right).

The percent values of the SBBs' sizes when overlapping all intervals are listed in

Table 10 (All SBB) as well as the percent values for overlapping of all main intervals (Main SBB). Furthermore, the values for each main interval with its adjacent interim diagrams, which are influenced by this main diagram (see section 4.2), are presented in this table. An example for such a combination of main and surrounding interim diagrams is the interval sequence [4:00,6:40], [6:40,9:20], [9:20,12:00] with main interval [6:40,9:20], (cf. Figure 11).

Sector	0	1	2	3
All SBB(%)	41.5	21.1	55.7	78.1
Main SBB (%)	41.5	21.2	66.7	79.4
[0:00-4:00]	48.1	44.2	69.2	82.4
[4:00-12:00]	85.8	45.7	66.7	81.3
[12:00-20:00]	81.1	45.3	55.8	87.7
[20:00-24:00]	41.5	42.9	88.7	95.6

Table 10: Example of Size of Sector Building Blocks in percent using interim diagrams

Some of the values for the Main SBB are higher than for the version using all SBBs, because the interim diagrams did not fully overlap the original shape of the main sectorisation. This is caused by optimising the interim diagrams as well instead of using the calculated intermediate vertices related to the surrounding main diagrams without adaptation. Without the optimisation, the interim diagrams would have been much more like intermediate steps and easier to handle and to understand by airspace controllers. However, it was shown in (Gerdes, Temme and Schultz, 2016) that the optimisation was able to improve the sectorisation in relation to task load and task load standard deviation substantially. Therefore, a balance has to be found between a sector shape easily applicable by controllers and a sectorisation with good values for the parameters of the evaluation function. The results for each main interval plus the surrounding interim diagrams are always better than those for the average main intervals themselves. This demonstrates that the interim diagrams are able to include the SBB of the belonging main interval and to change only smoothly in the direction of the next main interval, which is the desired result.

Figure 20 shows the course of the size of the building blocks for each sector with increasing number of included time intervals in relation to the size of the overlay of all time intervals. It can be seen that this value is stable over many time intervals and the reduction between time steps (size of new SAM) is around ten percent at most. This proves that the introduction of interim diagrams leads to a set of stable building blocks and to sharable airspace modules of reasonable size which can be handled by controllers. Nevertheless, the degree of free optimisation for the interim diagrams steered by the parameter group $w_{vc} \cdot E_{vc}$ of the evaluation function representing the vertex closeness (see section 4.3) has to be tested in simulation trials in the future, as it is already scheduled in current and upcoming research projects.

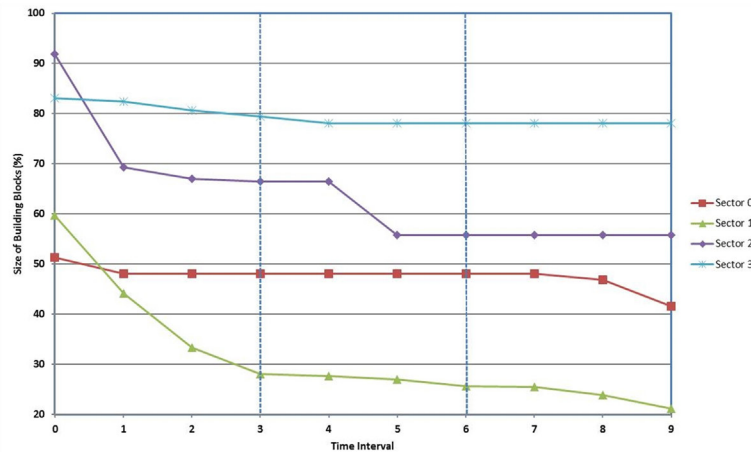


Figure 20: Evolution of Sector Building Blocks for each sector in percent of its size summarised over actual time interval and all preceding time intervals. Vertical lines indicate main intervals.

7 Conclusions and outlook

With our AutoSec approach, in particular the concepts of fuzzy clustering, Voronoi diagrams and evolutionary algorithms we developed an efficient approach to handle dynamic air traffic movements over the day of operations. Thus, we provide an appropriate airspace structure, which is generated automatically and considers the task load of airspace controllers (level and variation). AutoSec can be seen as an enabling technology for future flight-centric operations and multi-criteria optimisation.

We introduced our three-step approach of fuzzy clustering of traffic flows on the day of operations, generation of new sector structure based on Voronoi diagrams and application of evolutionary algorithm in order to adjust and optimise the new sector structure on dynamic air traffic demands. Therefore, we sufficiently solved the problem of non-convex boundaries and a smooth transition between two sectorisations for different air traffic demand. Within our developed simulation framework, we analysed a traffic sample of EDYYDUTA (EDYY Deco Sectors, Maastricht) area to allow a future comparison of our results with results conducted by Sergeeva et al. (2015). In particular, we addressed the optimisation of EDYYDUTA area based on the real sector structure and on our DAS approach.

The results are achieved by the simulation of real and optimised sectorisations (based on real sectors) as well as a completely new sectorisation approach (based on dynamic traffic demand), which allowed a comparison of the actual situation against the opportunities arising from dynamic, automated and optimised re-structuring. In the first case of optimising real sector structures, the application of the evolutionary algorithm improved the sectorisation (in particular task load variation) whilst staying close to the original shape of sectors. From an operational perspective, optimisation of existing sectorisations with regard to dynamic air traffic situations keeps additional workload for controllers low during transition between changing sectorisations. Using our AutoSec approach for initial sectorisation and optimisation, we found similar airspace structures for a four-sector environment with less-complex shapes (reduced number of vertices) and task load values on the same order of magnitude. Since the simplified shapes resulted in reduced design flexibility, the variance of task load increased. At this point, we have to mention that dependencies between real traffic samples and real sector shapes have not been taken into account. Furthermore, we demonstrated that AutoSec could be used as a substitute to provide (near-real) operational airspace structures for situations where no prior sectorisation exists. In addition, the detailed analysis of interim diagrams indicates that building blocks can be established which can be re-used to hold the work load for controllers at a reasonable level when changing from one sectorisation to the next.

Current research uses AutoSec to provide an operational platform for ecologically efficient operations (e.g. contrail avoidance) and urban airspace management. In the future, we will use AutoSec as a demonstration environment for a real controller environment, focussing on both operational requirements of controllers and future airspace/air traffic.

References

- Basu, A., Mitchell, J. and Sabhnani, G. (2009). Geometric algorithms for optimal airspace design and air traffic controller workload balancing. *Journal of Experimental Algorithmics*, 14 (2.3).
- Bezdek, J. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum Press.
- Bloem, M. and Gupta, P. (2010). Configuring airspace sectors with approximate dynamic programming. In: 27th International Congress of the Aeronautical Sciences (ICAS), Nice: ICAS.

- Chen, Y., Bi, H. and Zhang, D. (2013). Dynamic Airspace Sectorization via improved genetic algorithms. *Journal of Modern Transportation*, 21 (2), pp. 117-124.
- Cho, J. and Yoon, Y. (2018). How to access the capacity of urban airspace: A topological approach using keep-in and keep-out geofence. *Transp. Res. Part C: Emerg. Technol.*, 92, pp 137-149.
- de Berg, M., Cheong, O., van Kreveld, M. and Overmars, M. (2008). *Computational Geometry, Algorithms and Applications*. Berlin Heidelberg: Springer.
- de Oliveira, J.V. and Pedrycz, W. (2007). *Advances in Fuzzy Clustering and its Applications*. Chichester: John Wiley & Sons.
- Delahaye, D., Schoenauer, M. and Alliot, J. (1998). Airspace sectoring by evolutionary algorithms. In: *IEEE International Congress on Evolutionary Computation*. Anchorage: IEEE.
- Deveci, M. and Demirel, N. (2018). Evolutionary algorithms for solving the airline crew pairing problem. *Computers & Industrial Engineering*, 115, pp. 389-406.
- Djokic, J., Lorenz, B. and Fricke, H. (2010). Air traffic control complexity as workload driver. *Transp. Res. Part C: Emerg. Technol.*, 18, pp 930–936.
- Durand, N., Gianazza, D., Gotteland, J.-B. and Alliot, J. (2015). *Metaheuristics for air traffic management*. Hoboken: John Wiley & Sons.
- Eurocontrol, FAA (2016). *Comparison of Air Traffic Management-Related Operational Performance: U.S./Europe 2015*. Washington / Brussels: FAA / European Commission / EUROCONTROL
- European Commission (2004). *Framework for the creation of the single European sky, Regulation (EC) No549/2004*. Brussels: European Union.
- Flener, P. and Pearson, J. (2013). Automatic Airspace Sectorisation: A survey. [online] Available at: <https://arxiv.org/abs/1311.0653v1>, [accessed September 2017].
- Fortune, S. (1986). A sweepline algorithm for Voronoi diagrams. In: *Proc. of the Second Annual Symposium on Computational Geometry*. New York: SCG, pp. 313-322.
- Gerdes, I., Klawonn, F. and Kruse, R. (2004). *Evolutionäre Algorithmen*. Wiesbaden: Vieweg.
- Gerdes, I., Temme, A. and Schultz, M. (2016). Dynamic Airspace Sectorization Using Controller Task Load. In: *6th SESAR Innovation Days (SIDs)*. Delft: SESAR.
- Gianazza, D. (2010). Forecasting workload and airspace configuration with neural networks and tree search methods. *Artificial Intelligence*, 174 (7-8), pp. 530-549.
- Islami, A., Sun, M., Chaimatanan, S. and Delahaye, D. (2017). Optimization of military missions impact on civilian 4D trajectories. In: *ENRI International Workshop on ATM/CNS (EIWAC)*. Tokyo: ENRI.
- Jägare, P., Flener, P., Pearson, J. (2013). Airspace sectorisation using constraint-based local search. In: *10th ATM Seminar*, Chicago: FAA / EUROCONTROL.
- Kaltenhäuser, S., Morlang, F., Luchkova, T., Hampe, J. and Sippel, M. (2017). Facilitating Sustainable Commercial Space Transportation Through an Efficient Integration into Air Traffic Management. *New Space*, 5 (4), pp. 244-256.
- Keller, A. (2002). *Objective Function based Fuzzy Clustering in Air Traffic Management*. PhD. Otto-von-Guericke University Magdeburg, Germany.
- Kölker, K. and Lütjens, K. (2015). Using Genetic Algorithms to Solve Large-scale Airline Network Planning Problems. *Transportation Research Procedia*, 10, pp. 900-909.
- Kopardekar, P., Bilimoria, K. and Sridhar, B. (2007). Initial concepts for dynamic airspace configuration. In: *7th AIAA Aviation Technology, Integration and Operations Conference (ATIO)*. Belfast: AIAA.
- Kreuz, M., Luchkova, T. and Schultz, M. (2016). Effect Of Restricted Airspace On The ATM System. In: *WCTR Conference 2016*. Shanghai: WCTRS.
- Kulkarni, S., Ganesan, R. and Sherry, L. (2011). Static Sectorization Approach to Dynamic Airspace Configuration Using Approximate Dynamic Programming. In: *Integrated Communications, Navigation and Surveillance Conference (ICNS)*. Herndon: ICNS.
- Li, J., Wang, T., Savai, M. and Hwang, I. (2010). Graph-based algorithm for dynamic airspace configuration. *Journal of Guidance, Control and Dynamics*, 33, pp. 3-31.
- Luchkova, T., Vujasinovic, R., Lau, A., Schultz, M. (2015). Analysis of Impacts an Eruption of Volcano Stromboli could have on European Air Traffic. In: *11th ATM Seminar*. Lisbon: EUROCONTROL / FAA.
- Mehadhebi, K. (2000). A methodology for the design of a route network. In: *3rd ATM Seminar*. Napoli: EUROCONTROL / FAA.
- Meinecke, M. (2014). *Entwicklung und Evaluation von Lotsenarbeitsbelastungsmodellen in einer Schnellzeitsimulationsumgebung*. Master. DLR Braunschweig, Germany.
- Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin Heidelberg: Springer.
- Rosenow, J., Fricke, H. and Schultz, M. (2017). Air Traffic Simulation with 4D multi-criteria optimized trajectories. In: *Proc. of the 2017 Winter Simulation Conference*. Las Vegas: WCS Foundation, pp. 2589-2600.
- Schultz, M., Gerdes, I., Standfuß, T. and Temme, A. (2017). Future airspace design by dynamic sectorization. In: *ENRI International Workshop on ATM/CNS (EIWAC)*. Tokyo: ENRI.
- Schultz, M. (2017). Dynamic change of aircraft seat condition for fast boarding. *Transp. Res. Part C: Emerg. Technol.*, 85, pp. 131–147.
- Sergeeva, M., Delahaye, D., Zerrouki, L. and Schede, N. (2015). Dynamic Airspace Configuration Generated by Evolutionary Algorithms. In: *34th Digital Avionics Systems Conference*. Prague: DASC.
- Sergeeva, M., Delahaye, D., Mancel, C. and Vidosavljevic, A. (2017). Dynamic airspace configuration by genetic algorithm. *Journal of Traffic and Transportation Engineering*, 4(3), pp. 300-314
- SESAR Joint Undertaking (2015). *European ATM Master Plan*, <https://www.atmmasterplan.eu>.
- Sherali, H.D. and Hill, J.M. (2011). Configuration of airspace sectors for balancing air traffic controller workload. *Annals of Operations Research*, 203 (1), pp. 3-31.
- Soolaki, M., Mahdavi, I., Mahdavi-Amiri, N., Hassanzadeh and R., Aghajani, A. (2012). A new linear programming approach and genetic algorithm for solving airline boarding problem. *Applied Mathematical Modelling*, 36 (9), pp. 4060–4072.

- Standfuß, T., Gerdes, I., Temme, A. and Schultz, M., 2018. Dynamic Airspace Optimization. CEAS Aeronautical Journal.
- Sunil, E., Hoekstra, J., Ellerbroek, J., Bussink, F., Nieuwenhuisen, D., Vidosavljevic, A. and Kern, S. (2015). Metropolis: Relating Airspace Structure and Capacity for Extreme Traffic Densities. In: 11th ATM Seminar. Lisbon: EUROCONTROL / FAA.
- Tang, J.J., Alam, S., Lokan, C. and Abbass, H.A. (2012). A multi-objective approach for Dynamic Airspace Sectorization using agent based and geometric models. *Transp. Res. Part C: Emerg. Technol.*, 21 (1), pp. 89–121.
- Temme, A. and Helm, S. (2016). Unmanned Freight Operations. In: DLRK 2016, Braunschweig, Germany.
- Xue, M. (2009). Airspace Sector Redesign Based on Voronoi Diagrams. *Journal of Aerospace Computing, Information, and Communication*, 6 (12), pp. 624-634.
- Yangzhou, C. and Defu, Z. (2014). Dynamic airspace configuration method based on a weighted graph model. *Chinese Journal of Aeronautics*, 27 (4), pp. 903–912.
- Zelinski, S. and Lai, C. (2011). Comparing methods for dynamic airspace configuration. In: 30th Digital Avionics Systems Conference. Sydney: DASC.
- Zou, X., Cheng, P., An, B. and Song, J. (2016). Sectorization and Configuration Transition in Airspace Design. *Mathematical Problems in Engineering* 2016, 4, pp. 1-21.